Finding Anomalies in Network System Logs with Latent Variables

Kazuki Otomo
The University of Tokyo
otomo@hongo.wide.ad.jp

Satoru Kobayashi
NII
sat@hongo.wide.ad.jp

Kensuke Fukuda
NII/Sokendai
kensuke@nii.ac.jp

Hiroshi Esaki
The University of Tokyo
hiroshi@wide.ad.jp

ABSTRACT

System logs are useful to understand the status of and detect faults in large scale networks. However, due to their diversity and volume of these logs, log analysis requires much time and effort. In this paper, we propose a log event anomaly detection method for large-scale networks without pre-processing and feature extraction. The key idea is to embed a large amount of diverse data into hidden states by using latent variables. We evaluate our method with 15 months of system logs obtained from a nation-wide academic network in Japan. Through comparisons with Kleinberg's univariate burst detection and a traditional multivariate analysis (i.e., PCA), we demonstrate that our proposed method detects anomalies and ease troubleshooting of network system faults.

CCS CONCEPTS

• **Networks** → **Network management**;

KEYWORDS

Network log analysis; Latent variable analysis; Variational autoencoder;

ACM Reference Format:
Kazuki Otomo, Satoru Kobayashi, Kensuke Fukuda, and Hiroshi Esaki. 2018. Finding Anomalies in Network System Logs with Latent Variables. In *Big-DAMA'18: ACM SIGCOMM 2018 Workshop on Big Data Analytics and Machine Learning for Data Communication Networks , August 20, 2018, Budapest, Hungary.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3229607.3229608

1 INTRODUCTION

System logs are one of the most useful sources to understand the state of a network. In an operational network, syslog is widely used for collecting network logs and allows one to gather logs from all devices at one server. However, it is not easy for network operators to investigate the details of network problems with the logs because of their diverse and massive nature. Many log analysis methods for finding anomalies and their root causes have been proposed to overcome this problem [1, 9, 11, 12, 14, 18]. In many cases, one first classifies the logs by their message type (i.e., *log template*) then treats them as statistical time series to be later processed through statistical analysis. System logs have an intrinsic nature that makes automatic analysis very difficult; their time series show a variety of characteristics such as periodicity, burstiness, randomness, and sparseness, compared with other time series data. Most studies have been devoted to developing effective pre-processing methods (e.g., filtering, smoothing, denoising) and appropriate discriminative log time series features for analyzing their own data. However, we require enough domain knowledge about the underlying networks for optimizing the analysis methods.

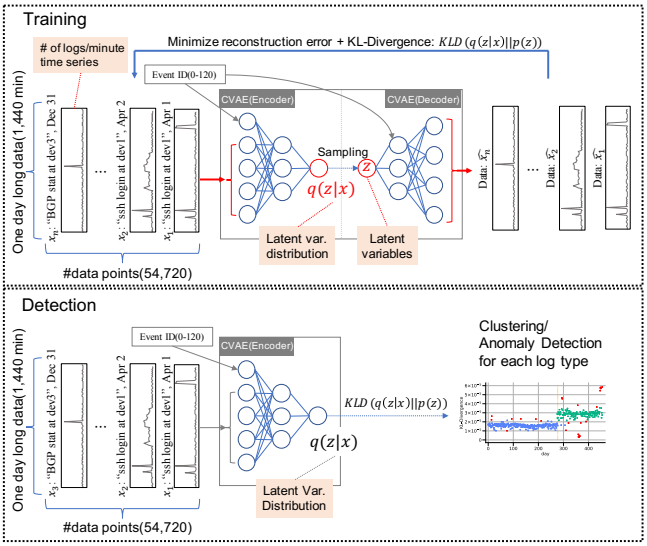

Figure 1: Analysis overview

We define an anomaly as time series behavior deviating from the normal or usual state in log time series (details are given in §3.3). To detect these anomalies, we propose a robust statistical method that handles complicated system log time series, without any specific (case-by-case) preprocessing (e.g., filtering, smoothing, auto-regression) for log anomaly detection. The key idea of our approach is to embed a large amount of diverse data into hidden states by using latent variables, also known as topic modeling or Latent Dirichlet Allocation (LDA) often used in natural language

Big-DAMA'18, August 20, 2018, Budapest, Hungary

ACM ISBN 978-1-4503-5904-7/18/08…$15.00
https://doi.org/10.1145/3229607.3229608

processing. A topic model discovers topics as latent variables from a collection of documents. We borrow the idea of this approach: by assuming hidden states in each log time series, we represent log time series features by latent variables. As a result, we can characterize each time series without any labels or preprocessing.

Our proposed method consists of two phases (see also Figure 1): (1) The training phase embeds log time series characteristics into latent variables by using Conditional Variational Autoencoder (CVAE). To mitigate the difficulty in handling various time series from devices together, we provide CVAE with type of log messages as a conditional label. (2) The detection phase highlights anomalies deviating from the distribution of the latent variables with clustering algorithms.

Using syslog data collected in a nation-wide academic network in Japan (SINET4), we confirm through evaluation that CVAE has higher discriminative power for analyzing complex time series than Kleinberg's univariate burst detection [8] and a traditional multivariate analysis (Principal Component Analysis; PCA). With case studies, we demonstrate that our method is useful in troubleshooting network system faults.

The contribution of this paper is twofold. We first propose our latent variable-based method for highly diverse log time series data without any data-specific preprocessing. Next, we discuss the effectiveness of the method when we applied it to real network syslog data.

2 RELATED WORK

Many studies have been conducted for finding anomalies and their root causes [1, 12, 14] in log data. Zhong et al. [18] proposed an anomaly detection method for both device and network errors with fine log time series feature creation. Kimura et al. [5] introduced an online failure prediction method based on log time series features. Lu et al. [11] focused on the task duration time and proposed root cause analysis methods with distributed computing system logs. As these methods are based on data specific feature creation, they perform well in each considered environment. However, to apply these methods to other network systems, we have to re-define features to optimize these methods. This requires deep domain knowledge of the underlying network systems to make efficient use of these methods. In addition, these methods miss unknown or new anomalies, which are not captured by the pre-defined features. Instead, we focus on an approach that does not require pre-definition of anomalies but learns the normal state of the system from log data.

There are methods of giving new insights for operators with knowledge mining from log data [4, 17]. Kobayashi et al. [9] proposed a time series causal inference method in network logs. Hacker et al. [3] introduced a log classification method based on the severity of network operation. These methods do not require predefined features because the features characterizing the data can be obtained through learning. However, in contrast with our method, they do not aim at anomaly detection but knowledge mining.

We focus on anomaly detection without specific feature creation for more general analysis of logs. PCA is a standard algorithm of learning or obtaining time series features from data. Lakhina et al. [10] proposed a general traffic analysis method based on feature learning with PCA. They successfully detected traffic anomalies without specific definitions of those anomalies. However, as log data are highly sparse and discrete, which differ with traffic data, it is not enough to learn features with such a naive approach (i.e., PCA). Thus, we use CVAE, which is based on a higher assumption that observed data are subject to latent variables. We choose a PCA based method as a baseline algorithm and aim to outperform it with our proposed method.

3 METHODOLOGY

3.1 Overview

The key idea of our proposed method is to model log time series characteristics based on the latent variables. Latent variable analysis is conducted to attempts to explain observable data by unobservable (i.e., latent) variables. In our context, as the observable data are log time series, we intend to represent the characteristics of observable log time series, such as periodicity, frequency, burstiness, by latent variables. By applying a clustering algorithm or anomaly detection method to the latent variables, we can distinguish anomalous behavior of log time series from the normal behavior. Formally, as it is difficult to compute latent variables directly, we have to estimate them by assuming certain statistical models. In this study, we rely on Conditional Variational Autoencoder (CVAE) for estimating latent variables.

An overview of the proposed method is illustrated in Figure 1. First, we construct the time series of the number of log appearances per device per template for one day. We call it a *data point*. Next, with CVAE, we reduce the dimensions of the original vectors by using CVAE. We call such a reduced vector a latent variable. We train CVAE model so that the latent variable effectively represents the potential behavior of log time series. Then, each data point is embedded into a latent variable. After that, we apply a clustering algorithm to the latent variables and finally detect outliers in log time series as anomalies.

3.2 Training phase

We briefly explain CVAE, a variation of the Variational Autoencoder (VAE). The VAE [7] is a stochastic variational inference method based on the variational Bayesian approach. Since this is an unsupervised method, annotations for logs are not necessary.

Let us consider one-day long log time series $\mathbf{x}$ generated by a random process based on invisible continuous random variable $\mathbf{z}$. This $\mathbf{z}$ is also subject to a prior distribution $p_\theta(\mathbf{z})$. Thus, $\mathbf{x}$ is subject to a conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$. The goal is to obtain latent variables $\mathbf{z}$ from input $\mathbf{x}$. In order to get the latent variables, we estimate the distribution $p_\theta(\mathbf{z}|\mathbf{x})$ with Bayes' theorem and approximate distribution $q_\phi(\mathbf{z}|\mathbf{x})$. Now, the objective is to maximize the marginal likelihood (MLH) $\log p_\theta(\mathbf{x}) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) + \mathcal{L}(\theta, \phi, \mathbf{x})$. This equation is rearranged as follows (details are given in [6, 7]).

$$\log p_\theta(\mathbf{x}) \geq \mathcal{L}(\theta, \phi, \mathbf{x}), \tag{1}$$

$$\mathcal{L}(\theta, \phi, \mathbf{x}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]. \tag{2}$$

We train the model that generates $q_\phi(\mathbf{z}|\mathbf{x})$ from input $\mathbf{x}$ by optimizing the lower bound $\mathcal{L}(\theta, \phi, \mathbf{x})$. The training process is as

```
sshd[21151]: Invalid user admin from 192.168.0.10
sshd[22569]: Invalid user virus from 192.168.0.15
                          ↓
          sshd[ * ]: Invalid user * from *
```

Figure 2: An example of log templates

follows. First, we consider the prior distribution as a Gaussian $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$ and let $q_\phi(\mathbf{z}|\mathbf{x})$ be a multivariate Gaussian. Next, we estimate the parameters (ϕ) of the posterior distribution ($q_\phi(\mathbf{z}|\mathbf{x})$) with a fully connected neural network (encoder). Then, we acquire the distribution of the latent variable $q_\phi(\mathbf{z}|\mathbf{x})$, and through the sampling process from it, we obtain the latent variable $\mathbf{z}$. Finally the decoder neural network reconstructs $\hat{\mathbf{x}}$ from the latent variable $\mathbf{z}$, and we feedback the loss values in Eq. (2). Reviewing Eq. (2), we can consider the first term (Kullback-Leibler divergence, KLD) as the distance between estimated distributions $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{z})$, and the second term (Marginal Likelihood, MLH) as the reconstruction error between $\mathbf{x}$ and $\hat{\mathbf{x}}$. Through these processes, we can compute Eq. (2) and proceed with the training of the encoder and decoder neural networks.

The CVAE [6] is a variation of the VAE. CVAE uses label information when generating the latent variable $\mathbf{z}$ and reconstructing $\hat{\mathbf{x}}$. We define the conditional label as the event that is an index of a unique combination of devices and log templates.

3.3 Detection phase

After applying CVAE to log time series data, we obtain the latent variables for each data. Now, as the latent variables well-describe time series trends, the goal of this phase is to build upon the time series description by latent variables to find the deviation from "normal" states.

Some latent analysis methods define the anomaly level based on reconstruction errors [10, 16]. In our case, we can use MLH for reconstruction errors. However, there are two problems with using MLH. (1) The reconstruction process is stochastic and MLH values follow a Gaussian distribution. (2) MLH is biased by the intensity of the original data.

To solve these problems, we rely on the distance between latent variable distribution ($q_\phi(\mathbf{z}|\mathbf{x})$) and the assumed prior distribution ($p_\theta(\mathbf{z})$) by computing $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$ in Eq. (2). After training, the learned model generates the distribution of the latent variable ($q_\phi(\mathbf{z}|\mathbf{x})$) from input data $\mathbf{x}$ as the output in a hidden layer (shown at the bottom of Figure 1). We use this distribution $q_\theta(\mathbf{z}|\mathbf{x})$ for detection, not the sampled value $\mathbf{z}$. Then, we deterministically compute the KLD ($D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$). The KLD represents the trend in the input time series $\mathbf{x}$. Thus, input data with similar KLD value belong to the same trend. Therefore, we can obtain reference behavior by clustering KLD values. Note that the KLD value is not an anomaly level but a state of input time series. Thus, we consider the data whose KLD value deviates from others as an anomaly regardless of the magnitude of numeric value of the KLD.

Figure 3 shows an example of the KLD and MLH values of the dataset. The horizontal axis shows data points in descending order of the number of log appearances, and the vertical axis shows

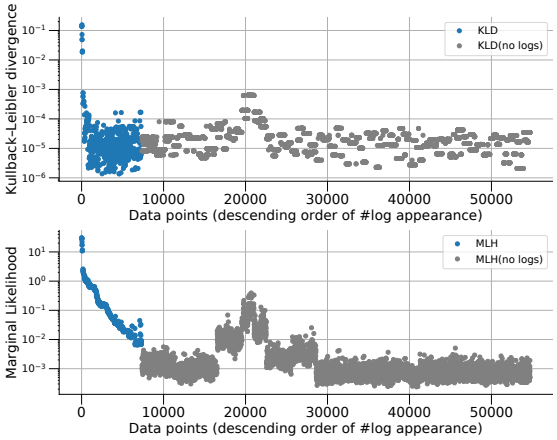

Figure 3: Kullback Leibler divergence and Marginal Likelihood (whole dataset)

the KLD (top) and MLH (bottom) values. In particular, the gray marks indicate days without any log appearances (i.e., 1,440-long zero vector), and the blue ones show more than equal to one log appearances on that day. Even if we input no log appearances data (i.e., gray marks in the Figure) to CVAE, KLD values are different for each event thanks to the weight of the conditional label (i.e., event ID). It enables such data points to belong to appropriate clusters depending on the behavior of the event. As shown in the figure, the MLH values are more spread and biased by the number of log appearances. In addition, they increase along with the number of log appearances. On the other hand, the KLD values are not biased by the number of log appearances and are more stable than MLH values. Therefore, we use the KLD as an indicator of time series trend.

Once latent variable distribution is obtained and the KLD is computed one can choose any clustering method for anomaly detection. In this study, we used a well-known density-based clustering algorithm, Density-Based Algorithm for Discovering Clusters (DBSCAN) [2]. By applying DBSCAN to the KLD time series for each event, we detect clusters of KLD (i.e., normal state of that type of log) and anomalies that do not belong to any clusters.

4 DATASET AND COMPARISON ALGORITHMS

4.1 Dataset

We use a set of network logs collected at SINET [15], a Japanese research and education network. This network connects over 800 academic organizations in Japan and consists of eight core routers, 50 edge routers, and 100 layer-2 switches. We use 456 days data from 2012/1/1 to 2013/3/31, consisting of ≈ 35M log messages from 130 devices (see also Table 1). This dataset is composed of well known format syslog data gathered from commodity L2 switches and L3 routers. Examples are shown in Figure 2 and the log category of the dataset are shown in Figure 4 (manually classified).

Table 1: SINET4 Dataset

	logs	devices	templates	events
All	34,722,578	130	1,789	9,226
Selected	943,257	24	33	120

Table 2: Comparison of methods

name	multivariate	determin.	linear trans.
burst detection	no	yes	-
PCA	yes	yes	yes
CVAE	yes	no	no

As log messages are string data, statistical techniques cannot be applied directly. Thus, we generate log templates from raw log messages with the supervised learning approach proposed by Kobayashi et al. [9]. Log templates are log messages without variables (e.g., IP address), as shown in Figure 2. We then classify logs into log templates and extract time series data from their time stamps for each template per device. We thus generate 1,789 unique log templates from the whole dataset. We define an *event* as 456 of 1-day log time series, which have the same log template in one device. As a preliminary analysis, we manually selected 120 events and applied CVAE to them (shown in Table 1) to make sure that the dataset has several anomalies and the diversity of the time series. We confirm that these selected events include a large variety of time series trends such as periodicity, burstiness, and sparseness.

We construct log time series of the number of log appearances for each minute per event per day (i.e., data point). The processing flow is as follows: After we classified raw log data into the type of logs per device, one event (for example, "*ssh login at device 1*") has 456 days long log entries. We first count the number of log appearances for each minute. Then, we split them for each days. We now have 456 data points of event "*ssh login at device 1*". In other words, each data point consists of one time series; It has 1,440 minute-length as X-axis and the number of log appearances as Y-axis (shown as x_1 and x_2 time series graphs in Figure 1). We also use conditional labels built on event labels concatenating log time series. To input event labels into CVAE, we first assign unique IDs to all the events and then encode them to one-hot vectors. If the "*ssh login at device 1*" event has ID i, the conditional label is a 120 long vector and values are zero except index i. The value of index i is one. Then, the conditional label has 120 dimensions. After repeating the same process to all the 120 events, we finally obtain $120 \times 456 = 54,720$ data points and conditional labels. In training and detection phase, each data point is an input for CVAE.

4.2 Baseline algorithms

To understand the effectiveness of CVAE for log anomaly detection, we compare it to two traditional baseline algorithms: Kleinberg's burst detection [8] and PCA. The three algorithms are summarized in Table 2.

4.2.1 Burst detection. Otomo et al. [13] conducted log analysis focusing on the burstiness and causality of log time series. They first removed trivial logs (e.g., periodic and very frequent logs) to prevent detecting trivial bursts then detected log bursts with Kleinberg's burst detection [8]. Applying the same algorithm to our dataset, we confirm 448 log bursts out of all 54,720 data points. In the following comparison, we use these bursts as a part of reference of log anomalies.

4.2.2 PCA. To detect traffic anomalies, Principal Component Analysis (PCA) is a well-studied algorithm [10, 16]. It translates a set of input traffic time series data into main and residual subspaces with a linear transformation. This algorithm is similar to CVAE mapping raw data to the latent space, though CVAE is a non-linear transformation. We implement a PCA-based anomaly detector. Applying PCA to a set of log time series, we obtain principal components of the input dataset and separate them into main and residual principal components with a threshold based on their variance coverage. We then reconstruct each time series with the main components and compute the reconstruction errors with the original data. Finally, we apply a clustering method (DBSCAN) to these reconstruction errors and obtain anomalies.

5 EVALUATION

In this section, we discuss the evaluation of our method using 456 day-long system log data obtained from SINET4. We compare the performance of our method with PCA and burst detection technique. Finally, we present two case studies that demonstrate the effectiveness of our method for network troubleshooting.

5.1 Parameter tuning

We first briefly describe the parameter tunings of CVAE. Encoder and decoder networks in CVAE each have four hidden layers. The input layer has 1,560 dimensions (1440 dimension of time series and 120 dimension of label data). The encoding neural network has four layers, and each layer has 512, 256, 128 and 64 units in order from the input layer. We set the latent variables dimensions to 10. The decoding neural network has the opposite structure of the encoder, but the output size is 1,440 so that the output is a reconstructed time series. To avoid over fitting, we dropout 30% of units for each layer during training. During training, we empirically set the number of epochs to 50 based on test trial results. We confirm that loss values converge after training. Next, we compute KLD for each time series using the learned encoder network and apply DBSCAN to KLD for each event. We tune DBSCAN parameters for each event so that a size of cluster is longer than a week. As the training process is stochastic, the results are not the same even if the loss values are converged. To mitigate this uncertainty, we train the model ten times and adopt anomalies detected in majority votes.

We use a commodity computer (Xeon CPU and GTX-1080 GPU) for processing. One training takes 489.5s on average of 10 trials. The detection phase requires 45.1s on average to calculate KLD and conduct DBSCAN.

5.2 Result overview

The results are summarized in Table 3. CVAE and PCA detected a similar number of anomalies; 1,759 anomalies with CVAE and 1,825 with PCA. The overlapping number of anomaly was 1,335. Note that a detected anomaly means time series behavior deviating from the normal state as defined in §1.

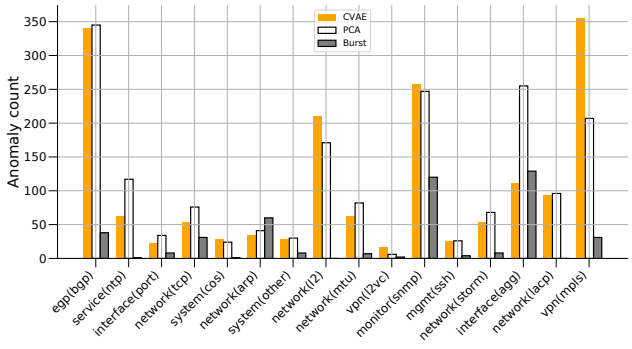

Figure 4: Number of anomalies per log category

Table 3: Summary of results

	Data	Anomalies	Precision	#Bursts
CVAE	54,720	1,759	84.6%	240 (53.6%)
PCA	54,720	1,825	69.5%	323 (72.1%)

We consider such anomalies as having a relationship to actual network system anomalies. To evaluate the effectiveness of detected anomalies for network management, we manually inspected all the detected anomalies and annotated them as true or false based on the SINET4 trouble ticket data and our domain knowledge for network management; For example, "*ssh login*" logs have periodic appearance because they are generated by a CRON process including remote ssh access. In this case, we consider periodic appearance as "normal" and lack of periodicity as "anomaly".

As shown in the table, CVAE's precision ($TP/(TP+FP)$) outperforms PCA's. We also confirm that CVAE and PCA detected more than half of the bursts. By inspecting the missing bursts, we find two parameter tuning issues: (1) when similar burst patterns last for a few days, DBSCAN extracts a cluster, not an outlier, the algorithm thus does not detect this burst as an anomaly. (2) Intensive bursts cause CVAE and PCA to detect a relatively small burst as a normal.

Figure 4 is the histogram of anomalies per log category. Orange bars indicate anomalies detected with CVAE, white ones with PCA, and gray ones with burst detection. In most cases, we find that CVAE correctly detects anomalies such as burstiness and lack of periodicity. Focusing on the case in which PCA detects more anomalies than CVAE, we find anomalies not detected with CVAE as false positives due to noise, which CVAE properly avoids (e.g., interface(agg) and service(ntp)). On the other hand, when CVAE detects more anomalies than PCA (e.g., vpn(mpls)), we find that they are mostly true positives thanks to its robustness against outliers (see § 5.3).

5.3 Detailed comparison

Anomalies hidden in periodicity: Figure 5 shows the detailed results of periodic remote access logs for 456 days. The top graph shows KLD computed with CVAE and the bottom graph illustrates reconstruction errors obtained with PCA. These logs appear once per hour due to an automated remote monitoring script. Some non-periodic bursts also occur in the logs. We confirm two significant outliers in the figure due to many log appearances (label (a)) and missing periodic logs (label (b)). As non-periodic anomalies represent failures of automatic remote access login, we intend to detect these non-periodic anomalies without detecting periodic bursts. As shown in this figure, CVAE and PCA correctly separate these anomalous data from others. Furthermore, the burst detection finds only periodic bursts. CVAE and PCA correctly learn event-wise features (in this case, periodic appearance) thanks to their discriminative power.

Sensitivity to phase changes: The periodic patterns of this log appearances (i.e., its phase) changes at the 274th day (label (c)). This phase change is due to the timing shift of a log appearing with the same frequency; this log appears in five minutes periods. From day 0 to day 273, each log appears almost at the same time and period (beginning at 0:44 in an hour period). After day 274, the appearance time shifts though the appearance period is the same (beginning at 0:10 in an hour period).

As shown in the bottom graph of Figure 5, it is difficult for PCA to detect the state change at the orange line because the values of reconstruction error seems stable.

On the other hand, CVAE easily detects the change in the behavior at the orange line and DBSCAN correctly separates the data at this point. Therefore, CVAE is more discriminative to capture the complex change in time series, which is not detected with PCA.

Robustness against outliers: KLD is robust against the intensity (i.e., number of log appearances) of the original data, as shown in Figure 3. On the other hand, we confirm that PCA's reconstruction errors (not shown in the figure) are also biased to the intensity, similar to the case with MLH.

Figure 6 (MPLS path down event) shows an example of CVAE's sensitivity to a small number of log appearances. The top graph shows KLD with CVAE, and the bottom graph shows the reconstruction errors with PCA. The solid blue circles are normal points and the crosses are anomalies based on DBSCAN clustering. Note that anomalies in the top graph are clustering results after merging ten trials, but the KLD values are a result of one trial. These data have one large outlier (around day 17; label (a)) which has a larger number of log appearances than other days. As shown in the figure, PCA is affected by this outlier, and many anomalous points are not detected. With CVAE, however, the KLD values are relatively spread compared with reconstruction errors, and DBSCAN detects other anomalies.

If there are no outliers but logs are noisy, PCA yields many false anomalies due to its noise-sensitivity. However, as CVAE is stochastic and merges the multiple results, we can mitigate the effect of noises. Therefore, CVAE exhibits better robustness to outliers and noises than PCA.

However, we also confirm that less than 1% of the data have high KLD values ($> 10^{-4}$), as shown in Figure 3. It is still possible that the clustering process may miss small KLD value changes due to these high values. We will work to improve this issue by tuning the clustering process as future work.

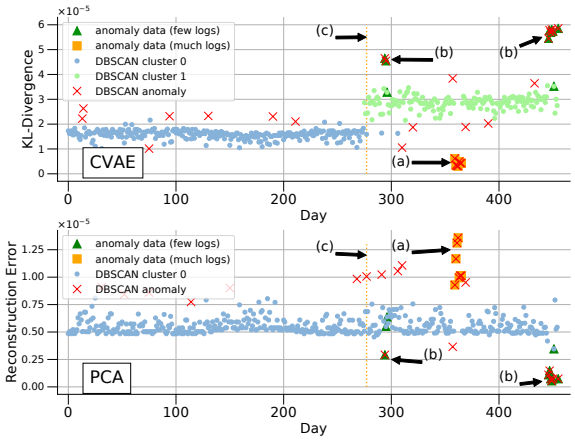

Figure 5: KLD (CVAE) and reconstruction errors (PCA)

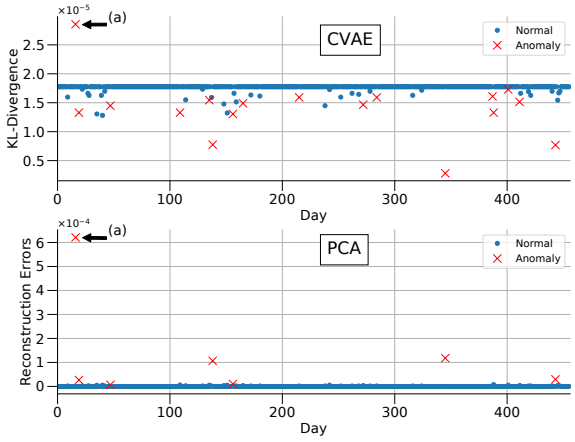

Figure 6: Robustness of CVAE

5.4 Case Studies

We now discuss anomaly examples detected from our proposed method.

5.4.1 BGP state change. Figure 7 shows the first example of detected anomalies. In this figure, each mark shows the KLD of the BGP state change logs from a router per day. Almost all days have a baseline ($\approx 6.0 \times 10^{-6}$). The detected anomalies in < 100 days are due to changes in network configuration. Furthermore, checking the raw logs around the 300th day (label (a)), we find that a connection with one particular AS suddenly became unstable. This failure had been also reported in a trouble ticket. PCA fails to detect most of these anomalies. CVAE, on the other hand, successfully discovers them.

5.4.2 MTU. The next example is MTU reduced logs from a router. This log template appears in a few days with two specific variables: multicast or unicast IP addresses. Figure 8 shows the KLD results. The blue round circles consisting of a baseline indicate no logs on those days. The orange crosses represent multicast logs,

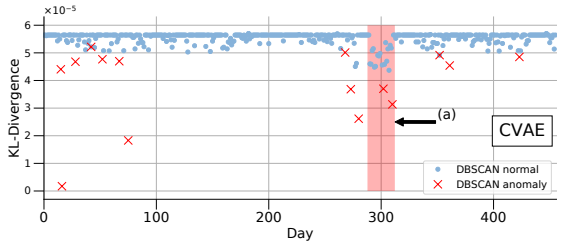

Figure 7: Case 1: BGP neighbor state change logs

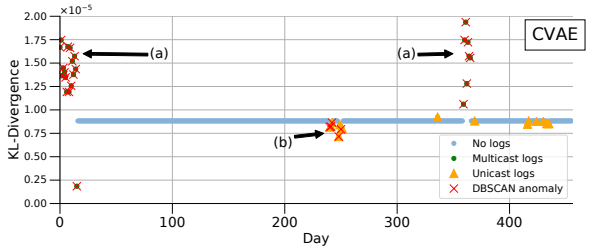

Figure 8: Case 2: MTU reduced logs in a router

and green crosses mean some unicast logs appear on those days. Compared with the baseline, the multicast logs show higher KLD values (cf. label (a)) and the unicast ones lower (label (b)) ones as anomalies. Thus, CVAE correctly separates these two types of anomalies even in this unique log template. We find that PCA misses most anomalies annotated by label (a).

6 CONCLUSION

We proposed a log analysis method based on the latent variable model to detect network system anomalies. We used Conditional Variational Autoencoder (CVAE) and applied a clustering algorithm to log time series with KL-divergence of the latent variable distribution. We confirmed that this method correctly works for highly sparse and diverse time series, which cannot be handled with a traditional PCA-based method. In addition, our case studies showed that some detected anomalies are helpful in finding network troubles. Towards the deployment of our method to real network management, we will work on addressing two issues: (1) how to update the trained model to use upcoming syslog data, and (2) how many logs are sufficient for catching the normal state in a system.

7 ACKNOWLEDGEMENTS

The authors would like to thank the SINET operation team for providing the syslog and trouble tickets data. The authors would also like to thank Johan Mazel for his comments on this paper and Roberto Gonzalez for shepherding.

REFERENCES

[1] E. Baseman, S. Blanchard, and E. Zongzelimyuntedu. Relational Synthesis of Text and Numeric Data for Anomaly Detection on Computing System Logs. *In Proc. IEEE ICMLA'16*, pages 2–5, 2016.

[2] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *In Proc. ACM KDD'96*, pages 226–231, 1996.

[3] T. Hacker, R. Pais, and C. Rong. A markov random field based approach for analyzing supercomputer system logs. *IEEE Transactions on Cloud Computing*, pages 1–1, 2017.
[4] T. Kimura, K. Ishibashi, T. Mori, H. Sawada, T. Toyono, K. Nishimatsu, A. Watanabe, A. Shimoda, and K. Shiomoto. Spatio-temporal factorization of log data for understanding network events. *In Proc IEEE INFOCOM'14*, pages 610–618, 2014.
[5] T. Kimura, A. Watanabe, T. Toyono, and K. Ishibashi. Proactive failure detection learning generation patterns of large-scale network logs. *In Proc CNSM'15*, pages 8–14, 2015.
[6] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling. Semi-Supervised Learning with Deep Generative Models. pages 1–9, 2014.
[7] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. (Ml):1–14, 2013.
[8] J. Kleinberg. Bursty and Hierarchical Structure in Streams. *In Proc. ACM KDD'02*, pages 91–101, 2002.
[9] S. Kobayashi, K. Otomo, K. Fukuda, and H. Esaki. Mining causality of network events in log data. *IEEE TNSM*, 15(1):53–67, 2018.
[10] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. *In Proc. ACM SIGCOMM'04*, 34(4):219, 2004.
[11] S. Lu, B. Rao, X. Wei, B. Tak, L. Wang, and L. Wang. Log-based Abnormal Task Detection and Root Cause Analysis for Spark. *In Proc. IEEE ICWS'17*, 2017.
[12] M. Moh, S. Pininti, S. Doddapaneni, and T.-S. Moh. Detecting Web Attacks Using Multi-stage Log Analysis. *In Proc. IEEE IACC'16*, pages 733–738, 2016.
[13] K. Otomo, S. Kobayashi, K. Fukuda, and H. Esaki. An Analysis of Burstiness and Causality of System Logs. *In Proc. AINTEC'17*, pages 16–23, 2017.
[14] M. Shatnawi and M. Hefeeda. Real-time failure prediction in online services. *In Proc. IEEE INFOCOM'15*, pages 1391–1399, 2015.
[15] S. Urushidani, M. Aoki, K. Fukuda, S. Abe, M. Nakamura, M. Koibuchi, Y. Ji, and S. Yamada. Highly available network design and resource management of SINET4. *Telecommunication Systems*, 56(1):33–47, 2014.
[16] W. Xu, L. Huang, A. Fox, D. Patterson, M. I. Jordan, L. Huang, A. Fox, D. Patterson, and M. I. Jordan. Detecting Large-Scale System Problems by Mining Console Logs. *In Proc. ACM SOSP'09*, pages 117–131, 2009.
[17] Z. Zheng, Z. Lan, B. H. Park, and A. Geist. System log pre-processing to improve failure prediction. *In Prco IEEE DSN'09*, pages 572–577, 2009.
[18] J. Zhong, W. Guo, and Z. Wang. Study on network failure prediction based on alarm logs. *In Proc. ICBDSC'16*, pages 23–29, 2016.